International Journal of Power Control Signal and Computation(IJPCSC)
Vol 8. No.1 – Jan-March 2016 Pp.109-114
©gopalax Journals, Singapore
available at : www.ijcns.com
ISSN: 0976-268X

CUSTOMIZED PREFETCHING ALGORITHM FOR SAVING BATTERY ENERGY IN MOBILE CLOUD COMPUTING

Dr.P.Calduwel Newton
Department of Computer Science
Government Arts College, Ayyarmalai
Kulithalai-639 120,Karur-Dt.
Tamilnadu, India.
calduwel@yahoo.com

Mrs.T.Cynthia
Department of Computer Applications
Bishop Heber College (Autonomous)
Tiruchirapalli-620 017.
cynthiasamabi@gmail.com

ABSTRACT— Energy saving is an important research topic in mobile phones. Energy can be saved in mobile devices by energy aware coding in mobile application development, computation offloading, cloudlets and optimized data transmission from cloud to mobile phones and vice versa. The proposed algorithm improves cache hit rate in data transmission between cloud and mobile phones by predicting the next application that would be used by the user. Customized Prefetching Algorithm (CPA) can be implemented in an existing protocol AppATP, where more number of missing cache hit rates is an issue. By implementing CPA that is 80% accurate in predicting the application that would be used next will surely reduces missing cache hit rates and reduces energy wastages.

KEY WORDS— Computation Offloading, Energy Saving, Data Transmission, Prediction, Cache hit rate.

I.INTRODUCTION

Mobile devices like smart phones, wearable devices play a vital role in day to day life. Their portability and network connectivity makes life even simpler. Banking, health care, social media and learning can be achieved through the mobile devices at anywhere and anytime. Though there can be plenty of usages, mobile devices lack in resources like processing speed and memory. Mobile cloud computing helps in solving the resource constrains by separating the storage and processing from mobile devices and provides the storage and processing services in cloud. The resource intensive tasks are offloaded from mobile devices to cloud to save battery in mobile devices. ThinkAir, Clone Cloud, are few of offloading frameworks that offloads the resource consuming computation to cloud to save energy. Computation offloading[1] is suitable for heavy computational works and the implementation is affected by the instability of wireless networks. In computation offloading, the data transmission between mobile devices and cloud is a highly challenging process because of the intermittent and instability in bandwidth.

Bandwidth is inversely proportional to time required for downloading, it means, if the bandwidth is high, the time taken to download is less. PengShu et.al.,[2] said that energy consumption was inversely proportional to bandwidth. It means, if the bandwidth is high, the energy consumption for data transmission is less.

The recent prediction from Cisco[3] explained that the mobile traffic will increase tenfold by 2019 and more offloading will take place. Although computation offloading helps to save energy in mobile devices, the unpredictable nature of bandwidth sucks battery life in poor networking connections especially when data is transmitted from cloud to mobile devices and vice versa. A recent forecast [4] discussed that by the end of 2020, eighty percent of mobile data traffic will be from smart phones. So, energy efficient data transmission between cloud and mobile devices gains more attention.

To make Computation Offloading as energy efficient, the prefetching and delay tolerant applications give space to do flexible scheduling of data transmission. Most of the commonly used mobile applications are delay tolerant and prefetching. So, flexible scheduling of data transmission can be achieved. Social Network Service (SNS), mobile learning, mobile health care, web browsing, news applications, media streaming and online shopping are few examples of delay tolerant applications. These kinds of applications need an optimization technique that optimizes the queue backlog and energy savings in data transmission. Lyapunov Optimization[5] is used to stabilize queues while optimizing the energy consumption of mobile devices when data transmission (i.e. downloading) is taken place.

This paper is organized as follows. Section I shows an introduction to computation offloading. The review of related works in Section II and Problem Statement is situated in Section III. Results and Discussions are in Section IV and Section-V concludes this paper. Finally References are listed.

II.**RELATED WORKS**

The inspiration for this work comes from the computation offloading frameworks Cuckoo, Think Air that saves mobile devices' battery life. Battery drain is the vital issue for most of the mobile device users. Computation offloading frameworks offloads the resource intensive tasks to the cloud and saves energy significantly. Many researchers implement these frameworks with little or no modifications. Energy consumption is measured by energy profilers[6]. Many researches find that energy unaware coding, data transmission between mobile devices and cloud [7], server selection and using too many profilers in offloading frameworks consumes lots of energy that turns offloading as obsolete. So, energy efficient frameworks[8] try to reduce energy consumptions in the above mentioned areas.

Energy consumption in data transmission grabs the researchers focus because most of the commonly used mobile applications need internet connection. Wi-Fi, 3G, 4G, bandwidth, cloud and mobile devices are frequently used terms in mobile cloud computing. Optimal scheduling for data transmission between mobile and cloud saves energy up to 25% to 30%[1] for prefetched and delay tolerant applications. The proposed work is inspired by an Energy Conserving Adaptive Mobile-Cloud Transmission Protocol(AppATP)[9] that prefetches frequently used data with minimum energy consumption for data transmission.

The researchers focus on Machine Learning(ML) algorithms and data mining to predict user's behavior[10] on application usage. Many works have been done on predicting user's behavior about the next application which is going to be used in the mobile, the application is going to be downloaded[11], the average time of application usage[12] by a user and the pattern of mobile user's active time. These studies help in a way to improve cache hit rate for prefetching, enhancing user experience and save time.

III.PROBLEM STATEMENT

AppATP is a protocol that optimizes energy savings for mobile devices during data transmission by implementing Lyapunov Optimization technique for prefetching mobile cloud applications. The protocol combines different SNS applications like Whatsapp, Twitter, Facebook etc. and makes it as a single application called MixSNS. It implements OAuth ,an open source protocol to retrieve data to be transmitted from these SNS applications' servers. The retrieved data is stored in MySql database along with Application_ID. Whenever the bandwidth is high or the queue length of an application's data is high, MixSNS application's server pulls data from the database and transmits to the application's cache in mobile device that is limited. The problem is prefetching has more number of missing hit rates in mobile devices. If the data stored in cache (limited in size) has not been used for long time, it is cleared before the application uses. The energy consumed for data transmission is wasted. AppATP doesn't employ any ML techniques based on user behaviors to improve cache hit rates.

This work proposes a Customized Prefetching Algorithm (CPA) that employs ML techniques in mobile side, based on user behavior to improve cache hit rates. Popularity based application list is sent to the cloud. Given a set of users$\{U_1,U_2,U_3\ldots.U_n\}$ and a set of prefetching type mobile cloud applications$\{A_1,A_2,A_3,\ldots A_n\}$, the problem is finding the applications list $\{AL_{U1},AL_{U2},AL_{U3}\ldots AL_{Un}\}$, that has the highest probability of being used by users. Here application list $AL_{U1} = \{A_1U_1,A_2U_1,A_3U_{1,}\ldots, A_nU_1\}$ i.e. application Ai for Ui where i={1,2,3,4…n}. This algorithm helps to prefetch data from cloud based on the frequent use of application and the time spent on applications by user.

The prefetching applications that are stored in a database table named Application_Details. Application_ID and Application_Name are stored. Another table called Application_User_Log contains Application_ID, Starting time of an application A_{ST} and Ending time of an application A_{ET} that have to be continuously stored whenever the application is opened by the user. Before prefetching the application's data from database, the cloud server sends request to mobile device to get the user priorities of applications. After receiving the request from cloud, the algorithm fetches the data that has been opened in the past 24 Hours.

The number of times the application has been opened, A_{DF}(Daily Frequency) and the number of seconds the application has been used A_{DU}(Daily Usage) are calculated. A_U(Application Usage in Seconds) is calculated by subtracting A_{ET-} $A_{ST.}$ The total number of seconds the application that has been used A_{DU} is calculated by summing up the A_U for all the occurrences of the applications. Sort the Application_User_Log table in Descending order based on A_{DF} and A_{DU}. Then the Application_ID's are sent to the cloud in same order. The cloud server fetches data from database based on the order of Application_ID's from mobile device and this improves the hit rate of cache. In Figure.1, the proposed work in is shown.

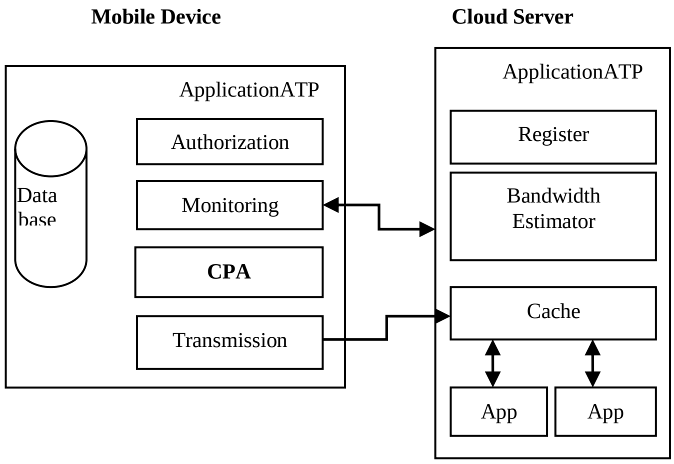

FIGURE.1.CPA IN MOBILE DEVICES

Required: Whenever an application is opened in mobile, the Application_ID and Start_Time(A_{ST}), have to be stored. Whenever an application is closed, the End_Time(A_{ET}) is also stored. Table 1. Shows the proposed CPA.

TABLE 1. Proposed CPA Algorithm

Input: Cloud Request for user priority of applications.
Output: Prioritized Application_ID's list.

Steps:

1. When the cloud sends request for application list A_{Ui}, the request time A_{RT} is considered to find the applications that has been opened and used in the past 24 Hours by checking ($A_{ET} < A_{RT}$ and $A_{ST} > A_{ET}$) for A_{RT} -24 .
2. For each application $_{Ai}$, calculate the application's Daily Frequency $_{ADF}$ by

summing up the number of times that application that has been opened.

3. For each application A_i calculate the usage A_U in seconds by $A_U = A_{ET} - A_{ST}$
4. For each application A_i, calculate the application's Daily Usage A_{DU}, by summing up the A_U..
5. Sort the list AL_{Ui} in descending order by A_{DF},
6. If two or more applications have same A_{DF} , consider A_{DU} and send the Application_IDs to the cloud.

In the performance evaluation of this algorithm, the A_{DF} and A_{DU} is calculated for 5 days. But 2 days datum is considered. From the Day1 result, AL_{U1} is obtained. Then AL_{U1} for the second day is calculated. Considering the top 5 applications from AL_{U1}(Day-1) and AL_{U1}(Day-2), accuracy is calculated by

$$\frac{\text{No.of applications in ALU1(Day2) that matches with the ALU1(Day1)}}{\text{No.of applications in ALU1 for Day1}} X100.$$

In spite of all the differences in user behavior, CPA shows good percentage of improvement in cache hit rates. In future, this algorithm would be fine tuned with many ML algorithms to find the optimum prediction list.

IV. RESULTS AND DISCUSSIONS

To test accuracy of popularity based ML technique for prefetch friendly applications, AppUsage Tracker, Quality Time and My Data Manager applications are installed in android based mobile phones. 15 users data are considered. App Usage Tracker and Quality Time applications tracks all the application's daily usage and daily frequency and gives the result in csv(comma separated values) format. The sample screen shot of applications are shown in Figure. 2.

FIGURE 2. SCREENSHOTS OF THE APPLICATIONS

This work considers 24 hours as comfortable time slot. From the data collected for this work, the average daily usage of internet through mobile devices are calculated as 2 Hours and 45 minutes. A popular survey shows[12], Indian spend on average of 169 minutes in mobile devices. The data shows that, WhatsApp, Facebook, Gmail, Chrome, YouTube, News Channels and Twitter are popularly used by the users. So, they are taken for this work. Indians spent 47% of their time in SNS applications[13] and this work agrees with the global statistics. Based on the study, there is not much variation in the daily usage and daily frequency of applications between timeslots and data usage on applications impact prediction.

Popularity based ML technique suggests same application list which are mostly used by all the users. But, users use different applications based on their needs. So, Popularity based ML techniques are applied based on each and every user's daily usage and daily frequency. This work saves the application's name, A_{ST} and A_{EN} in the following format shown in Table.2.

TABLE 2.APPLICATION DETAILS OF PARTICULAR USER

App Name	***Start Date & Time A_{ST}***	***Raw Duration (Seconds)*** $A_U = A_{EN} - A_{ST}$
Facebook	3/3/2016 22:44	7
Facebook	3/3/2016 22:44	3
Facebook	3/3/2016 22:44	16
System UI	3/3/2016 22:45	1
WhatsApp	3/3/2016 22:47	2

WhatsApp	3/3/2016 22:47	4
WhatsApp	3/3/2016 22:47	4
WhatsApp	3/3/2016 22:47	2
WhatsApp	3/3/2016 22:47	2

"SELECT app_id, count(app_id) as occurrences,SEC_TO_TIME(SUM(TIME_TO_SEC(TIMEDIFF(end_time,start_time)))) AS total_time FROM `app_time` WHERE end_time < NOW() AND start_time > DATE_SUB(NOW(), INTERVAL 24 HOUR) GROUP BY app_id ORDER BY occurrences, total_time DESC".

The sample outputs for two users are given in Table.3 and Table.4.

TABLE 3. DAILY FREQUENCY AND DAILY USAGE OF U1

AL_{U1}:(Day1)			
S. No.	*APP NAME*	*A_{DF}*	*A_{DU} (Secs.)*
1	WhatsApp	442	4821
2	Facebook	99	2355
3	Quality Time	15	181
4	Gmail	12	124
5	AppUsage Tracker	12	80

AL_{U1}:(Day2)			
S. No.	*APP NAME*	*A_{DF}*	*A_{DU} (Secs.)*
1	WhatsApp	293	5452
2	Facebook	81	2652
3	PlayStore	4	73
4	AppUsage tracker	5	70
5	Gmail	2	35

TABLE 4. DAILY FREQUENCY AND DAILY USAGE OF U2

AL_{U2}:(Day1)			
S. No.	*APP NAME*	*A_{DF}*	*A_{DU} (Secs.)*
1	WhatsApp	56	3561
2	AppUsage Tracker	38	951
3	Quality Time	25	1274
4	Whatsdog	23	413
5	Facebook	17	4454

AL_{U2}:(Day2)			
S. No.	*APP NAME*	*A_{DF}*	*A_{DU} (Secs.)*
1	WhatsApp	69	3284
2	Quality Time	16	2461
3	AppUsage Tracker	15	281
4	Gmail	12	221
5	Facebook	10	1554

When the top 5 or top 6 applications are considered, this algorithm gives 80% accurate result. When the number of days increased, there are chances for more accurate predicting system. This algorithm increases the cache hit rate in the existing system. The energy consumption for the MixSNS application is E means, this CPA algorithm helps to reduce the energy by ΔE. By implementing this algorithm in AppATP, the energy is E - ΔE.

V.**CONCLUSION**

The presented CPA helps in computation offloading techniques to avoid energy waste in more number of missed caches hit rates. The prior work doesn't include any techniques to study user behavior to improve cache hit rates. By implementing CPA in android OS based mobile phones, cache hit rates are improved and predicting user behavior is accurate by 80% and in future classification machine learning techniques can be embedded to predict more accurate. The energy wastage in the previous work is reduced by E - ΔE. Though it is a little overhead for mobile devices, the benefits perceived by the users are very much more than that.

REFERENCES

[1] Minhaj Ahmad Khan," A survey of computation offloading strategies for performance improvement of applicationlications running on mobile devices ",Journal of Network and Computer Applications, Volume 56,2015,pp 28-40.

[2] Peng Shu, Fangming Liu, Hai Jin, Min Chen, Feng Wen,Yupeng Qu and Bo Li, "eTime: Energy-Efficient Transmission between Cloud and Mobile Devices", INFOCOM, Proceedings IEEE, 2013, pp. 195-199.

[3]http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

[4] http://techcrunch.com/2015/06/02/6-1b-smartphone-users-globally-by-2020-overtaking-basic-fixed-phone-subscriptions/#.1j6fkoo:RPIH

[5] Weiwei Fang, Xiaoyan Yin , Yuan An , Naixue Xiong, Qiwang Guo and Jing Li, "Optimal scheduling for data transmission between mobile devices and cloud", Information Sciences, Volume 301, 2015,pp 169-180.

[6] K.Ravindranath, Dr. K. Raja Sekhara Rao," A Survey on Energy aware offloading Techniques for Mobile Cloud Computing", International Journal of Computer Trends and Technology (IJCTT), Volume 4 Issue 7, 2013,pp 2082-2086

[7] Xing Liu, Chaowei Yuan, Zhen Yang, and Enda Peng, "Wireless-Uplinks-Based Energy-Efficient Scheduling in Mobile Cloud Computing", Mathematical Problems in Engineering, Volume 2015, pp. 1-10.

[8] Salwa Adriana Saab, Ali Chehab and Ayman Kayssi, "Energy Efficiency in Mobile Cloud Computing: Total Offloading Selectively Works. Does Selective Offloading Totally Work?", International Conference on Energy Aware Computing Systems and Applications", 2013.

[9] Peng Shu ; John C. S. and Lui, "AppATP: An Energy Conserving Adaptive Mobile-Cloud Transmission Protocol", IEEE Transactions on Computers, Volume 64, Issue 11, 2015, pp. 3051 – 3063.

[10] Ke Huang, Xiaoxiao Ma, Chunhui Zhang and Guanling Chen, "Predicting Mobile Applicationlication Usage Using Contextual Information", Ubiquitous Computing, Pittsburgh, USA, 2012, pp. 1-7.

[11] Chang Tan, Qi Liu, Enhong Chen and Hui Xiong, "Prediction for Mobile Application Usage Patterns", Nokia MDC Workshop, Newcastle, UK, 2012.

[12] Ricardo Baeza-Yates,Di Jiang, Fabrizio Silvestri and Beverly Harrison, "Predicting The Next Application That You Are Going To Use", Web Search and Data Mining', Shanghai, China, 2015.pp 2–6.

[13] http://yourstory.com/2015/07/smartphone-user-persona-report-vserv/

[14]https://www.techinasia.com/india-web-mobile-data-jan-2015

[15]http://timesofindia.indiatimes.com/tech/tech-news/Indians-spend-47-of-their-time-on-applicationlications-like-WhatsApplication-Report/articleshow/47852496.cms